CLAIMS

What is claimed is:

1. A method for partitioning program modules, comprising:

providing affinity weights among the modules; wherein a relationship between two modules constitutes an affinity weight for those two modules;

based on the affinity weights among the modules,

providing a weight threshold; and

assigning a first module associated with an affinity weight that indicates the first module is most closely related to a second module; and

qualifying affinity weights that are associated with the first module, by comparing these affinity weights to the weight threshold; and

assigning, to the group, all modules that are associated with the affinity weights qualified in the qualifying step.

2. The method of claim 1 wherein an affinity weight in the step of qualifying is qualified based on one or a combination of the following logical relationship with the weight threshold: equal to, greater than.

3. The method of claim 1 further comprising the steps of:

a) qualifying affinity weights that are associated with the modules assigned to the group by the step of assigning, by comparing these affinity weights to the threshold; and

b) assigning, to the group, all modules associated with the affinity weights qualified in step a).

4. The method of claim 1 wherein an affinity weight for two modules of the program modules is provided based on one or more optimization opportunities between the two modules.

5. The method of claim 1 wherein the relationship between the two modules is based on one or a combination of:

a number of calls across the two modules;

a possibility for in-lining a function in a module of the two modules;

a characteristic of a call graph of functions in the two modules;

a frequency of a global variable referenced in the two modules;

a characteristic of a parameter passed between functions in the two modules;

a possibility for de-virtualizing a virtual function in a module of the two modules;

6. The method of claim 1 wherein:

an affinity weight for two modules of the program modules is provided by a formula $f_1w_1 + f_2w_2 + \ldots f_kw_k$;

each weight w_i is associated with a factor indicating a relationship between the two modules; and

each f_i is a weight percentage of the factor.

7. The method of claim 1 wherein the weight threshold is calculated using a total value of the affinity weights among the modules.

8. The method of claim 7 wherein the weight threshold is calculated using further a percentage value.

9. The method of claim 8 wherein the percentage value is derived from the capability of a compiler to handle a number of modules.

10. The method of claim 1 being implemented as program instructions embodied in a computer-readable medium.

11. A method for partitioning modules, comprising:

a) providing a weight threshold;

b) determining if there are modules remained to be partitioned,

if there is not, then stopping the method;

else proceeding to step c);

c) finding among the modules that have not been assigned to a group a module associated with the highest affinity weight among the affinity weights associated with the modules that have not been assigned to a group, and assigning this module to a new group;

d) for each module in the new group created in step c) that has not been processed,

identifying the each module as a first module;

iterating through each module neighboring to the first module; wherein a first module neighboring to a second module if the first module and the second module is related by an affinity weight;

if the neighboring module has not been assigned to a group, and an affinity weight between the neighboring module and the first module is qualified based on the weight threshold, then assigning the neighboring module to the new group; and

e) proceeding to step b).

12. The method of claim 11 wherein the affinity weight between the neighboring module and the first module is further qualified based on one or a combination of the following logical relationship: lesser than, equal to, greater than.

13. The method of claim 11 being implemented as program instructions embodied in a computer-readable medium.

14. A method for providing an affinity weight threshold for use in partitioning program modules, comprising:

providing a percentage value;

providing affinity weights among the modules;

providing a total value of the affinity weights;

using the percentage value and the total value of the affinity weights to provide a percentage of the total of the affinity weights;

using the percentage of the total of the affinity weights and a sum weight to provide the affinity weight threshold; the sum weight being the sum of at least two affinity weights.

15. The method of claim 14 wherein the percentage value is tunable based on the capacity of a compiler.

16. The method of claim 14 wherein the affinity weight threshold is provided when an affinity weight added to the sum weight causing the sum weight being one or a combination of:

equal to the percentage of the total of the affinity weights; and

greater than the percentage of the total of the affinity weights.

17. The method of claim 14 being implemented as program instructions stored in a computer-readable medium.

18. A method for providing an affinity weight between two modules for use in partitioning modules, comprising:

determining k factors; k being an integer number; each factor representing a distinct relationship between the two modules; and

providing a sum of f_iw_i as the affinity weight; the subscript i running k times;

wherein

each w_i is associated with a factor;

each f_i is a weight factor of a factor; and

a sum of f_i being equal to 100%.

19. The method of claim 18 wherein the relationship between the two modules is based on one or a combination of:

a number of calls across the two modules;

a possibility for in-lining a function in a module of the two modules;

a characteristic of a call graph of functions in the two modules;

a frequency of a global variable referenced in the two modules;

a characteristic of a parameter passed between functions in the two modules;

a possibility for de-virtualizing a virtual function in a module of the two modules.

20. A computer-readable medium embodying program instructions for performing a method for partitioning program modules, the method comprising:

a) providing affinity weights among the modules; wherein a relationship between two modules constitutes an affinity weight for those two modules;

b) based on the affinity weights among the modules,

providing a weight threshold; and

assigning a first module associated with an affinity weight that indicates the first module is most closely related to a second module; and

c) qualifying affinity weights that are associated with the first module, by comparing these affinity weights to the weight threshold; and

d) assigning, to the group, all modules that are associated with the affinity weights qualified in step c).

e) qualifying affinity weights that are associated with all modules assigned to the group by step d), by comparing these affinity weights to the threshold; and

f) assigning, to the group, all modules associated with the affinity weights qualified in step e).